Secure recharge of disposable RFID tickets

Riccardo Focardi Flaminia Luccio

Università Ca' Foscari, Venezia
{focardi,luccio}@unive.it

FAST 2011
15-16 September 2011, Leuven

Outline

RFID technology

- Radio Frequency IDentification (RFID) is more an more used
  1. transportation and logistics
  2. hospitals
  3. inventory
  4. passports
  5. ski resorts
  6. race timing
  7. animal identification
  8. human implant (!?)
  9. ...

RFID technology

- Radio Frequency IDentification (RFID) is more an more used

1. transportation and logistics
2. hospitals
3. inventory
4. passports
5. ski resorts
6. race timing
7. animal identification
8. human implant (!?)
9. ...

RFID tickets

- RFID cards for public transportation
- disposable cards
  1. chip-on-paper, very cheap
  2. one or more tickets of the same kind
  3. non-rechargeable
  4. minimal security mechanisms
- personal cards
  1. plastic, credit-card like cards
  2. tickets or different contracts on the same card
  3. rechargeable
  4. strong authentication mechanisms

How do they work?

- ISO/IEC 14443: very popular HF (13.56 MHz) standard for 'proximity cards' used for identification
  - proximity cards and proximity coupling device, i.e., the reader
- initialization and anticollision: the reader talks with exactly one card if more are in the field
- card API: the card implements a set of functions that can be invoked by the reader
- secure module: the reader may have a Security Access Module (SAM) to perform cryptographic operations and authentication
- security mechanisms: are specific for a certain card (and its API)

Mifare Ultralight cards

(a) Memory organization

Byte number →	0	1	2	3	Page
ID	ID0	ID1	ID2	Check1	0
ID	ID3	ID4	ID5	ID6	1
Check/Lock	Check2	Internal	Lock0	Lock1	2
OTP	OTP	OTP	OTP	OTP	3
Data	R/W	R/W	R/W	R/W	4
Data	R/W	R/W	R/W	R/W	5
Data	R/W	R/W	R/W	R/W	...
Data	R/W	R/W	R/W	R/W	15

(b) Lock bytes Lock0 and Lock1

Lock0	L_7	L_6	L_5	L_4	L_3	BL_1	BL_2	BL_{OTP}
Lock1	L_{15}	L_{14}	L_{13}	L_{12}	L_{11}	L_{10}	L_9	L_8

Security mechanisms

- very simple API: read/write of pages
- with some limitations:
  1. unique ID: read-only, included in encryptions or MACs to avoid card cloning
  2. OTP: One Time Programmable, monotone, prevents ticket reuse
  3. lock bytes: for read-only time-based contracts, e.g. skipass
  4. block locking bits: prevent locking of pages that need to be modified, e.g., the OTP
- no security API: security is built on top of the above mechanisms

$\Rightarrow$ security flaws found in real application, e.g., the OV-chipkaart in Amsterdam and Rotterdam.

Example (Card cloning)

Sample (flawed) code for consuming n tickets from a card

```
read(4 : 7);
if (MAC_K(p4, p6) = p7) then
    p4 := p4 − n;
    p5 := BUS_ID();
    p6 := TIMESTAMP();
    p7 := MAC_K(p4, p6);
    write(4 : 7);
```

0, 1	Id
2	Lock
3	Otp
4	R
5	D_1
6	D_2
7	$MAC_K(R, D_2)$

Example (Card cloning)

Sample (flawed) code for consuming n tickets from a card

```
read(4 : 7);
if (MAC_K(p4, p6) = p7) then
    p4 := p4 − n;
    p5 := BUS_ID();
    p6 := TIMESTAMP();
    p7 := MAC_K(p4, p6);
    write(4 : 7);
```

0, 1	Id
2	Lock
3	Otp
4	R
5	D_1
6	D_2
7	$MAC_K(R, D_2)$

0, 1	Id′
2	
3	
4	
5	
6	
7	

Example (Card cloning)

Sample (flawed) code for consuming n tickets from a card

```
read(4 : 7);
if (MAC_K(p4, p6) = p7) then
    p4 := p4 − n;
    p5 := BUS_ID();
    p6 := TIMESTAMP();
    p7 := MAC_K(p4, p6);
    write(4 : 7);
```

0, 1	Id
2	Lock
3	Otp
4	R
5	D_1
6	D_2
7	$MAC_K(R, D_2)$

$\xrightarrow{COPY}$

0, 1	Id′
2	
3	
4	R
5	
6	D_2
7	$MAC_K(R, D_2)$

Example (double-use)

Including the Id in the MAC prevents cloning

read(0 : 7);
if $(MAC_K(p_0, p_1, p_4, p_6) = p_7)$ then
  $p_4 := p_4 - n$;
  $p_5 := BUS\_ID()$;
  $p_6 := TIMESTAMP()$;
  $p_7 := MAC_K(p_0, p_1, p_4, p_6)$;
  write(4 : 7)

0, 1	Id
2	Lock
3	Otp
4	R
5	D_1
6	D_2
7	$MAC_K(\mathrm{Id}, R, D_2)$

Example (double-use)

Including the Id in the MAC prevents cloning

```
read(0 : 7);
if (MAC_K(p_0, p_1, p_4, p_6) = p_7) then
    p_4 := p_4 − n;
    p_5 := BUS_ID();
    p_6 := TIMESTAMP();
    p_7 := MAC_K(p_0, p_1, p_4, p_6);
    write(4 : 7)
```

0, 1	Id
2	Lock
3	Otp
4	R
5	D_1
6	D_2
7	$MAC_K(\mathrm{Id}, \mathrm{R}, D_2)$

$\xrightarrow{COPY}$

$\mathrm{R}, D_2, MAC_K(\mathrm{Id}, \mathrm{R}, D_2)$

Example (double-use)

Including the Id in the MAC prevents cloning

```
read(0 : 7);
if (MAC_K(p0, p1, p4, p6) = p7) then
    p4 := p4 − n;
    p5 := BUS_ID();
    p6 := TIMESTAMP();
    p7 := MAC_K(p0, p1, p4, p6);
    write(4 : 7)
```

0, 1	Id
2	Lock
3	Otp
4	$R - n$
5	D'_1
6	D'_2
7	$MAC_K(Id, R - n, D'_2)$

$\xrightarrow{COPY}$

use tickets $R, D_2, MAC_K(Id, R, D_2)$

Example (double-use)

Including the Id in the MAC prevents cloning

read(0 : 7);
if $(MAC_K(p_0, p_1, p_4, p_6) = p_7)$ then
  $p_4 := p_4 - n$;
  $p_5 := BUS\_ID()$;
  $p_6 := TIMESTAMP()$;
  $p_7 := MAC_K(p_0, p_1, p_4, p_6)$;
  write(4 : 7)

0, 1	Id
2	Lock
3	Otp
4	R
5	D_1
6	D'_2
7	$MAC_K(Id, R, D_2)$

$\xRightarrow{COPY}$

use tickets $R, D_2, MAC_K(Id, R, D_2)$

$\xLeftarrow{RESTORE}$

reuse the card arbitrarily

Example (the OTP)

Use the OTP to count resources on the card (initialized as $32 - R$)

read$(0:7)$;
if $(MAC_K(p_0, p_1, p_3, p_6) = p_7)$ then
  $p_3 := incOTP^n(p_3)$;
  $p_5 := BUS\_ID()$;
  $p_6 := TIMESTAMP()$;
  $p_7 := MAC_K(p_0, p_1, p_3, p_6)$;
  write$(3:7)$

0, 1	Id
2	Lock
3	Otp
4	
5	D_1
6	D_2
7	$MAC_K(\text{Id}, \text{Otp}, D_2)$

Example (the OTP)

Use the OTP to count resources on the card (initialized as $32 - R$)

read$(0:7)$;
if $(MAC_K(p_0, p_1, p_3, p_6) = p_7)$ then
  $p_3 := incOTP^n(p_3)$;
  $p_5 := BUS\_ID()$;
  $p_6 := TIMESTAMP()$;
  $p_7 := MAC_K(p_0, p_1, p_3, p_6)$;
  write$(3:7)$

0, 1	Id
2	Lock
3	Otp
4	
5	D_1
6	D_2
7	$MAC_K(\text{Id}, \text{Otp}, D_2)$

$\xrightarrow{COPY}$

$\text{Otp}, D_2, MAC_K(\text{Id}, \text{Otp}, D_2)$

Example (the OTP)

Use the OTP to count resources on the card (initialized as $32 - \mathrm{R}$)

read$(0:7)$;
if $(MAC_K(p_0, p_1, p_3, p_6) = p_7)$ then
  $p_3 := incOTP^n(p_3)$;
  $p_5 := BUS\_ID()$;
  $p_6 := TIMESTAMP()$;
  $p_7 := MAC_K(p_0, p_1, p_3, p_6)$;
  write$(3:7)$

0, 1	Id
2	Lock
3	Otp $+ n$
4	
5	D'_1
6	D'_2
7	$MAC_K(\mathrm{Id}, \mathrm{Otp} + n, \mathrm{D}'_2)$

$\overset{COPY}{\Longrightarrow}$

use $\quad \mathrm{Otp}, \mathrm{D}_2, MAC_K(\mathrm{Id}, \mathrm{Otp}, \mathrm{D}_2)$

Example (the OTP)

Use the OTP to count resources on the card (initialized as $32 - R$)

read(0 : 7);
if $(MAC_K(p_0, p_1, p_3, p_6) = p_7)$ then
  $p_3 := incOTP^n(p_3)$;
  $p_5 := BUS\_ID()$;
  $p_6 := TIMESTAMP()$;
  $p_7 := MAC_K(p_0, p_1, p_3, p_6)$;
  write(3 : 7)

0, 1	Id
2	Lock
3	Otp + n
4	
5	D_1
6	D'_2
7	$MAC_K(Id, Otp, D_2)$

$\xRightarrow{COPY}$ use $Otp, D_2, MAC_K(Id, Otp, D_2)$

$\xLeftarrow{RESTORE}$ FAIL!

Summary

- the unique ID, if in the MAC, prevents cloning
- the OTP can be used to store the number of tickets on the card
- monotonicity of OTP ensures that tickets cannot be reused

NOTE 1 cards cannot be recharged (the OTP is irreversible)

NOTE 2 cards cannot store different kind of tickets (one OTP)

NOTE 3 code can become complex leading to security flaws

motivated by the above issues

1. we propose a new way of using the OTP which admits recharges and different kind of tickets on the same card
2. we develop a typing-discipline to develop secure APIs on top of the insecure card APIs

Our proposal

- explicit resource counters, as in the first examples
- OTP incremented by 1 when (even n) resources are consumed
- the OTP does not represent the number of resources
  1. it is initialized as 0
  2. it is incremented at any 'irreversible event'

Our proposal

- explicit resource counters, as in the first examples
- OTP incremented by 1 when (even n) resources are consumed
- the OTP does not represent the number of resources
  1. it is initialized as 0
  2. it is incremented at any 'irreversible event'

read(0 : 7);
if $(MAC_K(p_0, p_1, p_3, p_4, p_6) = p_7)$ then
  $p_3 := incOTP(p_3)$;
  $p_4 := p_4 - n$;
  $p_5 := BUS\_ID()$;
  $p_6 := TIMESTAMP()$;
  $p_7 := MAC_K(p_0, p_1, p_3, p_4, p_6)$;
  write(3 : 7)

0,1	Id
2	Lock
3	Otp
4	R
5	D_1
6	D_2
7	$MAC_K(\mathrm{Id}, \mathrm{Otp}, \mathrm{R}, D_2)$

Our proposal

- explicit resource counters, as in the first examples
- OTP incremented by 1 when (even n) resources are consumed
- the OTP does not represent the number of resources
  1. it is initialized as 0
  2. it is incremented at any 'irreversible event'

read(0 : 7);
if $(MAC_K(p_0, p_1, p_3, p_4, p_6) = p_7)$ then
  $p_3 := incOTP(p_3)$;
  $p_4 := p_4 - n$;
  $p_5 := BUS\_ID()$;
  $p_6 := TIMESTAMP()$;
  $p_7 := MAC_K(p_0, p_1, p_3, p_4, p_6)$;
  write(3 : 7)

0,1	Id
2	Lock
3	Otp + 1
4	R − n
5	D'_1
6	D'_2
7	$MAC_K(Id, Otp + 1, R - n, D'_2)$

Type-based analysis

A linear type-and-effect system with judgement $\vdash \Gamma \ [\![c]\!] \ \Gamma'$

Type-based analysis

A linear type-and-effect system with judgement $\vdash \Gamma \ [\![c]\!] \ \Gamma'$

1. we model production and consumption of resources via special annotations *produce*(R), *consume*(R)

Type-based analysis

A linear type-and-effect system with judgement $\vdash \Gamma \ [\![c]\!] \ \Gamma'$

1. we model production and consumption of resources via special annotations *produce*(R), *consume*(R)
2. critical operations always performed after the MAC check

Type-based analysis

A linear type-and-effect system with judgement $\vdash \Gamma \; [\![c]\!] \; \Gamma'$

1. we model production and consumption of resources via special annotations *produce*(R), *consume*(R)
2. critical operations always performed after the MAC check

$$\frac{\begin{array}{c}\Gamma(p_i) = \mathsf{Mac}[\mathsf{i}_0, \ldots, \mathsf{i}_m] \qquad \Gamma(p_{\mathsf{i}_0}), \ldots, \Gamma(p_{\mathsf{i}_m}) = \bullet \\ \vdash \Gamma\{p_{\mathsf{i}_0} : \Gamma(\mathsf{i}_0), \ldots, p_{\mathsf{i}_m} : \Gamma(\mathsf{i}_m)\} \; [\![\mathsf{c}_1]\!] \; \Gamma' \qquad \vdash \Gamma \; [\![\mathsf{c}_2]\!] \; \Gamma'\end{array}}{\vdash \Gamma \; [\![\text{if } \mathit{MAC}_K(p_{\mathsf{i}_1}, \ldots, p_{\mathsf{i}_m}) = p_i \text{ then } \mathsf{c}_1 \text{ else } \mathsf{c}_2]\!] \; \Gamma'}$$

Type-based analysis

A linear type-and-effect system with judgement $\vdash \Gamma\ [\![c]\!]\ \Gamma'$

1. we model production and consumption of resources via special annotations *produce*(R), *consume*(R)
2. critical operations always performed after the MAC check

$$\frac{\Gamma(p_i) = \mathsf{Mac}[\mathsf{i}_0, \ldots, \mathsf{i}_m] \qquad \Gamma(p_{\mathsf{i}_0}), \ldots, \Gamma(p_{\mathsf{i}_m}) = \bullet \qquad \vdash \Gamma\{p_{\mathsf{i}_0} : \Gamma(\mathsf{i}_0), \ldots, p_{\mathsf{i}_m} : \Gamma(\mathsf{i}_m)\}\ [\![\mathsf{c}_1]\!]\ \Gamma' \qquad \vdash \Gamma\ [\![\mathsf{c}_2]\!]\ \Gamma'}{\vdash \Gamma\ [\![\text{if } \mathit{MAC}_K(p_{\mathsf{i}_1}, \ldots, p_{\mathsf{i}_m}) = p_i \text{ then } \mathsf{c}_1 \text{ else } \mathsf{c}_2]\!]\ \Gamma'}$$

3. production and consumption controlled via linear effects; OTP incremented and written back to the card before consumption

Type-based analysis

A linear type-and-effect system with judgement $\vdash \Gamma\ [\![c]\!]\ \Gamma'$

1. we model production and consumption of resources via special annotations *produce*(R), *consume*(R)
2. critical operations always performed after the MAC check

$$\frac{\Gamma(p_i) = \mathsf{Mac}[\mathsf{i}_0, \ldots, \mathsf{i}_m] \qquad \Gamma(p_{\mathsf{i}_0}), \ldots, \Gamma(p_{\mathsf{i}_m}) = \bullet \qquad \vdash \Gamma\{p_{\mathsf{i}_0} : \Gamma(\mathsf{i}_0), \ldots, p_{\mathsf{i}_m} : \Gamma(\mathsf{i}_m)\}\ [\![\mathsf{c}_1]\!]\ \Gamma' \qquad \vdash \Gamma\ [\![\mathsf{c}_2]\!]\ \Gamma'}{\vdash \Gamma\ [\![\text{if } \mathit{MAC}_K(p_{\mathsf{i}_1}, \ldots, p_{\mathsf{i}_m}) = p_i \text{ then } \mathsf{c}_1 \text{ else } \mathsf{c}_2]\!]\ \Gamma'}$$

3. production and consumption controlled via linear effects; OTP incremented and written back to the card before consumption

$$\frac{\Gamma(p_i) = \mathsf{R}}{\vdash \Gamma\ [\![p_i := p_i - n]\!]\ \Gamma, \mathsf{R}^n + \mathsf{W}_i} \qquad \frac{\mathsf{iOtp} \in \Gamma \qquad \mathsf{W}_3 \notin \Gamma}{\vdash \Gamma, \mathsf{R}^n\ [\![\mathit{consume}(\mathsf{R})^n]\!]\ \Gamma}$$

Typing the example

$$
\begin{array}{lll}
\vdash \Gamma & [\![\mathsf{read}(0:7); &]\!] \Gamma, \Gamma'' \\
\vdash \Gamma, \Gamma'' & [\![\mathsf{if}\ (\mathit{MAC}_K(p_0, p_1, p_2, p_3, p_4, p_6) = p_7)\ ]\!] & \\
\vdash \Gamma, \Gamma' & [\![\quad p_3 := \mathit{incOTP}(p_3); &]\!] \Gamma, \Gamma', \mathsf{iOtp}, \mathsf{W}_3 \\
\vdash \Gamma, \Gamma', \mathsf{iOtp}, \mathsf{W}_3 & [\![\quad \mathsf{write}(3) &]\!] \Gamma, \Gamma', \mathsf{iOtp} \\
\vdash \Gamma, \Gamma', \mathsf{iOtp} & [\![\quad p_4 := p_4 - n; &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{W}_4 \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_4 & [\![\quad p_5 := \mathit{BUS\_ID}(); &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5} \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5} & [\![\quad p_6 := \mathit{TIMESTAMP}(); &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5,6} \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5,6} & [\![\quad p_7 := \mathit{MAC}_K(p_0, p_1, p_2, p_3, p_4, p_6); &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5,6,7} \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5,6,7} & [\![\quad \mathsf{write}(4:7) &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp} \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp} & [\![\quad \mathit{consume}(\mathsf{R})^n; &]\!] \Gamma, \Gamma''
\end{array}
$$

Typing the example

$$
\begin{array}{lll}
\vdash \Gamma & [\![\mathsf{read}(0:7); &]\!] \Gamma, \Gamma'' \\
\vdash \Gamma, \Gamma'' & [\![\mathsf{if}\ (\mathit{MAC}_K(p_0, p_1, p_2, p_3, p_4, p_6) = p_7)\ ]\!] & \\
\vdash \Gamma, \Gamma' & [\![\quad p_3 := \mathit{incOTP}(p_3); &]\!] \Gamma, \Gamma', \mathsf{iOtp}, \mathsf{W}_3 \\
\vdash \Gamma, \Gamma', \mathsf{iOtp}, \mathsf{W}_3 & [\![\quad \mathsf{write}(3) &]\!] \Gamma, \Gamma', \mathsf{iOtp} \\
\vdash \Gamma, \Gamma', \mathsf{iOtp} & [\![\quad p_4 := p_4 - n; &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{W}_4 \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_4 & [\![\quad p_5 := \mathit{BUS\_ID}(); &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5} \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5} & [\![\quad p_6 := \mathit{TIMESTAMP}(); &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5,6} \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5,6} & [\![\quad p_7 := \mathit{MAC}_K(p_0, p_1, p_2, p_3, p_4, p_6); &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5,6,7} \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp}, \mathsf{W}_{4,5,6,7} & [\![\quad \mathsf{write}(4:7) &]\!] \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp} \\
\vdash \Gamma, \Gamma', \mathsf{R}^n, \mathsf{iOtp} & [\![\quad \mathit{consume}(\mathsf{R})^n; &]\!] \Gamma, \Gamma''
\end{array}
$$

i	Γ
0	Id
1	Id
2	Lock
3	Otp
4	R_T
5	Data
6	Data
7	Mac[0, 1, 2, 3, 4, 6]

	Γ'	Γ''
p_0	Id	•
p_1	Id	•
p_2	Lock	•
p_3	Otp	•
p_4	R_T	•
p_5	Data	Data
p_6	Data	•
p_7	Mac[0, 1, 2, 3, 4, 6]	Mac[0, 1, 2, 3, 4, 6]

Results

Attacker model

- any untrusted code not using K
- any well-typed program (API) possibly using K

For well-typed programs we formally prove that

1. any time a resource is consumed the OTP on the card has been incremented
2. after an arbitrary number of well-typed programs and attacker runs the number of resources R on the card is less than or equal to the number of resources recharged and not yet consumed, i.e., $\#produce(\mathrm{R}) - \#consume(\mathrm{R})$
3. cards with no valid MACs can never originate a *consume*(R)

Conclusion

- Mifare Ultralight cards
  - very cheap, implement minimal security mechanisms
  - application-level security is completely up to the programmer
- a new way of using the OTP enabling the recharge of cards and different kinds of tickets on the same card
- a type-checkable simple language to develop security APIs
  - applications can build on top of these type-checked APIs
- future work: we intend to implement a type-checker on real programming languages and test APIs on working applications (possible collaboration with public transport company in Venice)

References


Mifare ultralight contactless single-ticket IC, 2010.
Product data sheet. Rev. 3.8 22 December 2010.
Available at www.nxp.com/documents/data_sheet/MF0ICU1.pdf.


M. Bugliesi, S. Calzavara, F. Eigner, and M. Maffei.
Resource-aware Authorization Policies for Statically Typed Crypto Protocols.
In Proceedings of IEEE CSF'11.


M. Centenaro, R. Focardi, F. Luccio, and G. Steel.
Type-based analysis of PIN processing APIs.
In *Proceedings of ESORICS'09.*


P. Siekerman and M. van der Schee.
Security evaluation of the disposable ov-chipkaart v1.7, 2007.
Research Project for a Master Thesis, University of Amsterdam.


A. Tanenbaum.
Dutch public transit card broken, 2008.
Available at http://www.cs.vu.nl/~ast/ov-chip-card/.